Securing Kubernetes ingress gateway resource with Cert-manager-Atlas plugin

Cert-manager-ATLAS Issuer Securing Ingress Use Case

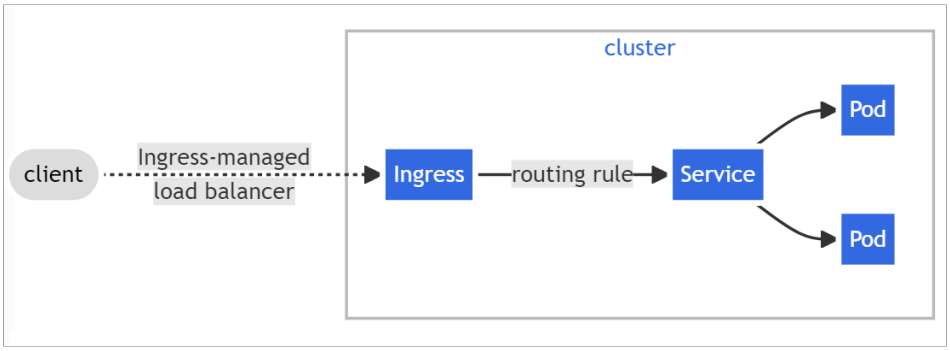
Pre-requisites:

1. AWS Account
2. Nginx Ingress
3. One Valid Domain Name
4. Kops and Kubectl
5. Helm Package Manager
6. Cert-manager & its CRD's
7. Cert-manager-Atlas Plugin

What is Ingress in Kubernetes?

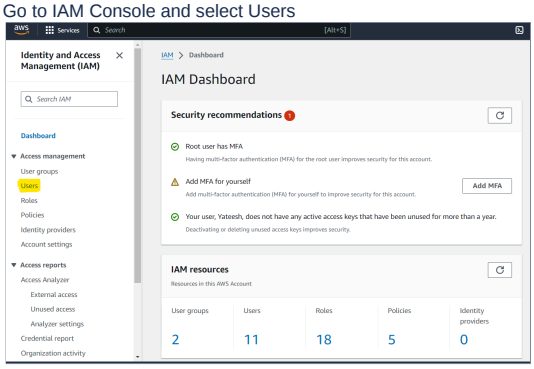
Ingress exposes HTTP and HTTPS routes from outside the Kubernetes cluster to services within the cluster. Traffic routing is controlled by network policy defined in the Ingress resource.

Here is a simple example where an Ingress sends all its traffic to one Service:

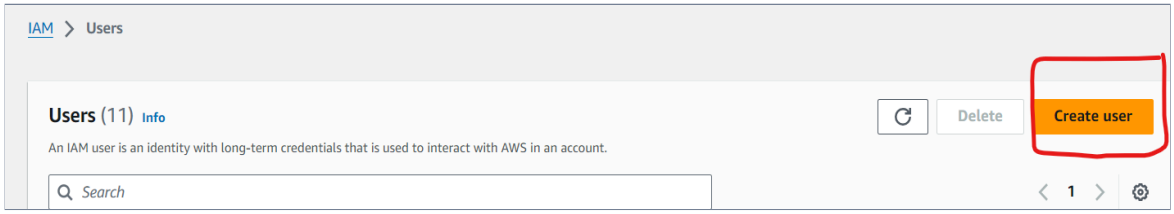


Steps to secure your Nginx-Ingress with GlobalSign's Trusted TLS certificate using Cert-manager-Atlas Plugin:-

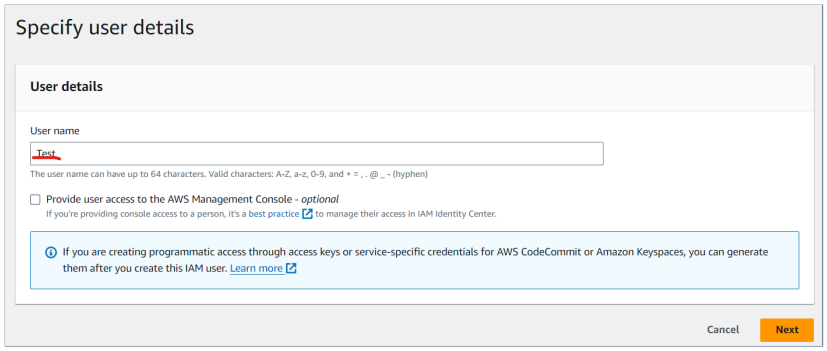
1. **Create AWS Instance and a user with an IAM Role**
    a. **AWS Ubuntu EC2 Instance - Use this AWS documentation for creating an Ubuntu Instance**
    b. **Create a User in the IAM Console with the required permissions**
        i. Go to IAM Console and select Users



ii. Create user

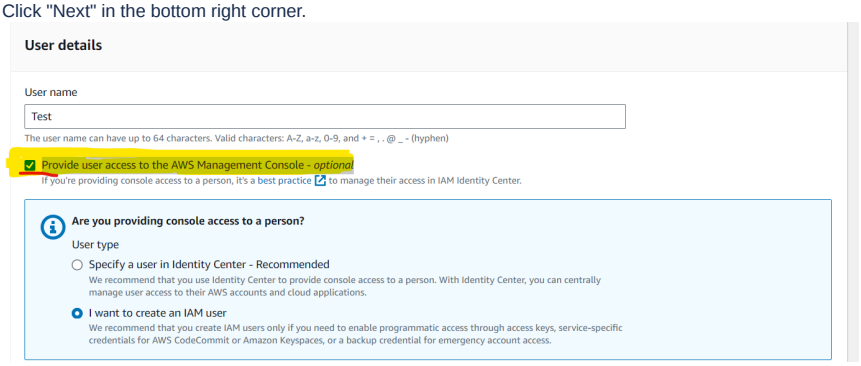


iii. Give a name to the user



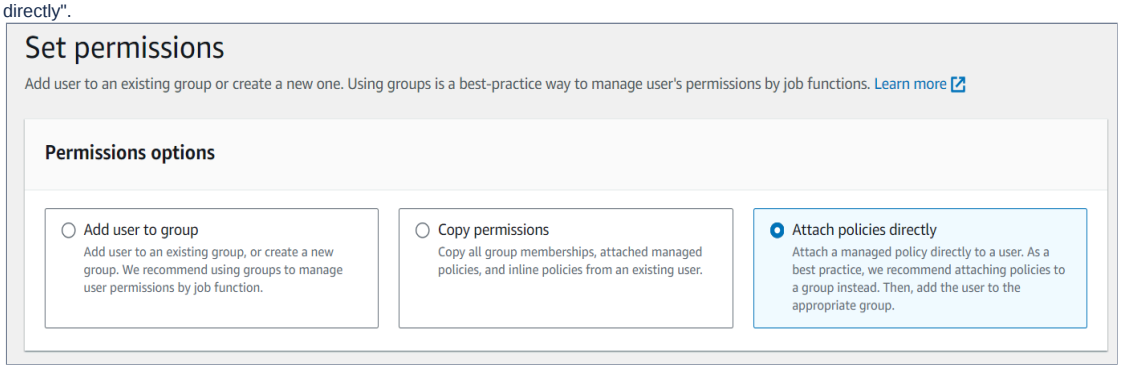
iv. Fill the checkbox of "Provide user Access to the AWS Management Console"

1. Select "I want to create an IAM user "
2. In Console Password, choose "Autogenerate Password" or "Custom Password" based on your choice.
3. Click "Next" in the bottom right corner.

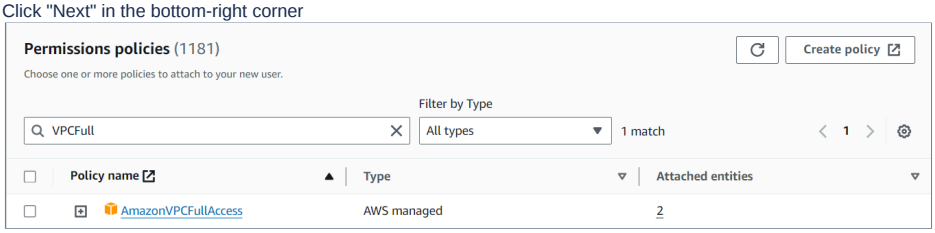


v. In Set Permissions, choose

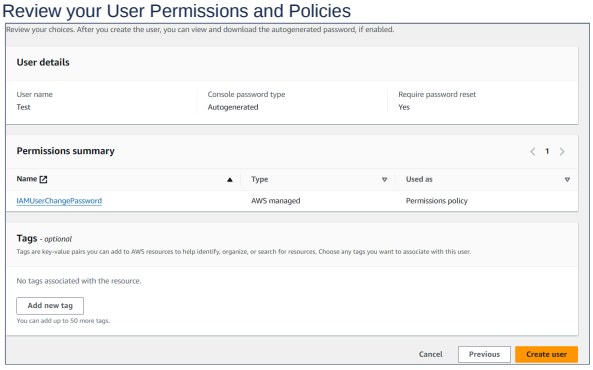
1. Select "Add User to a group" in case if you already have defined policies for a particular user group, otherwise choose "Attach Policies directly".



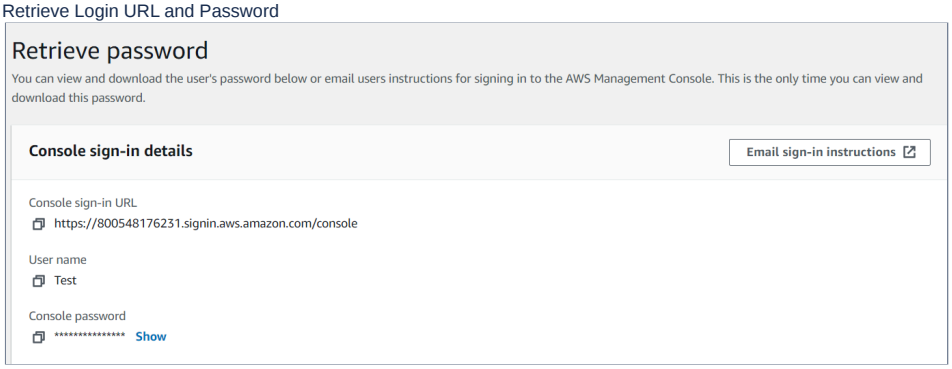
2. In Permission Policies, provide the following permissions to the user(Note:- You can provide permissions based on your own requirements as this is just for the example purposes.)
   a. VPCFullAccess
   b. EC2FullAcces
   c. S3FullAccess
   d. Route53FullAccess
   e. IAMFullAccess
3. Click "Next" in the bottom-right corner



4. Review your User Permissions and Policies

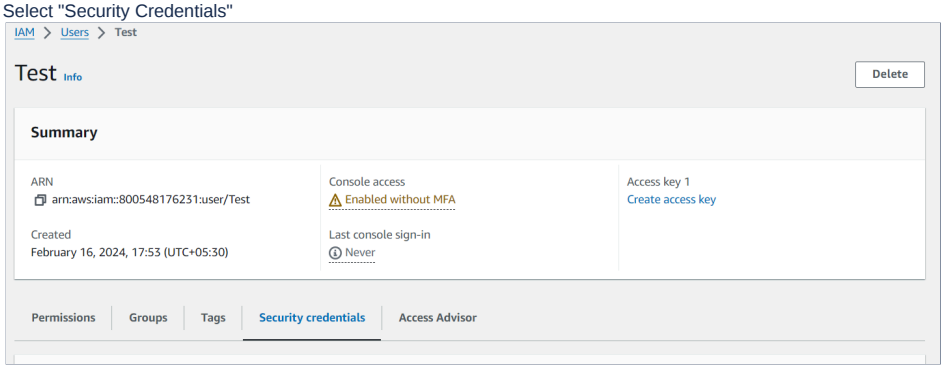


5. Select "Create User" in the bottom-right corner and your user will be created.
6. Retrieve Login URL and Password

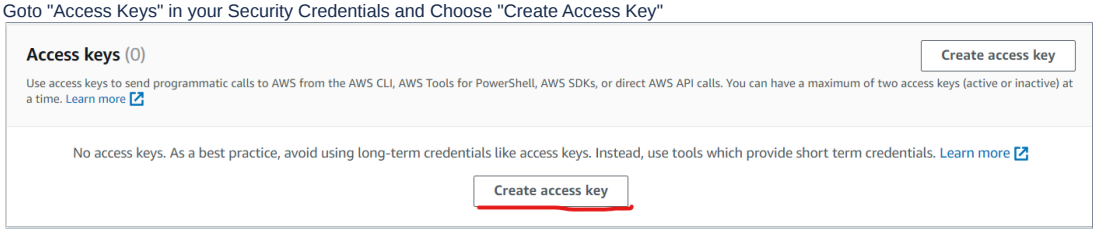


c. **Provide programmatic Access to the created user**

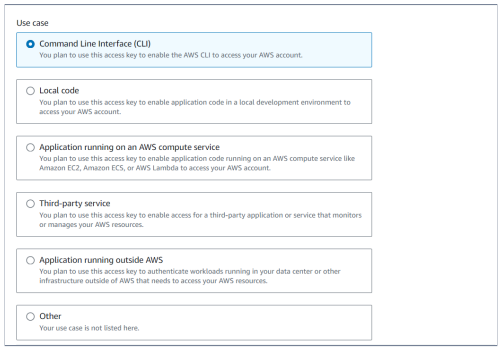
i. Goto IAM and then users again
ii. Select your created user
iii. Select "Security Credentials"



iv. Goto "Access Keys" in your Security Credentials and Choose "Create Access Key"

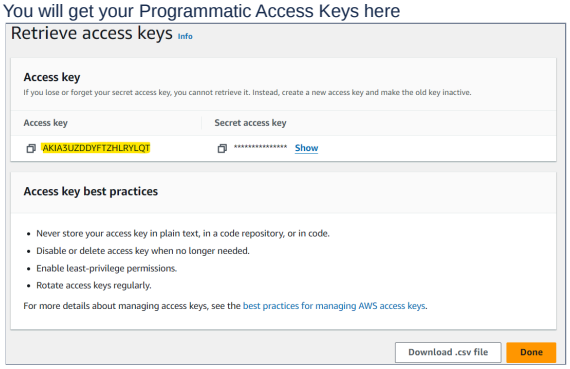


v. Goto the Use Case and select "AWS CLI"



vi. Click Next and then "Create Access Key"

vii. You will get your Programmatic Access Keys here



2. **Configure Kubernetes Cluster & Install Cert-manager & Its CRD's into your Instance**
   a. **Connect to your AWS ec2 instance which you have created in the Step-1 (Ref.)**
   b. **Once you are logged in to your instance, then Install the following tools**
      i. **Install Unzip**

```
$sudo apt install
```

      ii. **Configure AWS CLI**

```
$curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$unzip awscliv2.zip
$sudo ./aws/install
```

   c. **Now, Configure AWS CLI with the following commands and the programmatic access keys created in the above steps**

```
$aws configure
#enter the Access key ID and Secret access key.
#Provide the region details i.e., us-east-1 or any other
#Give output format as "json".
#Generate public and private keys
$ssh-keygen
```

   d. **Once AWS CLI is configured, then create one S3 Bucket with a name as per your choice(we have used "pki.atlasqa.co.uk") for storing the states of your Kubernetes Cluster.**

```
$aws s3api create-bucket --bucket pki.atlasqa.co.uk --region eu-west-1
```

   e. **After creating the S3 Bucket using AWS CLI, Create one hosted zone from Route53** (The name of the zone either same as bucket name or should be successor of bucket name for example if bucket name is example.com then hosted zone name should be abc.example.com).
   f. **Install Helm**

```
$curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$chmod 700 get_helm.sh
$./get_helm.sh
```

   g. **Install Kubectl and KOps**
      i. **Installing Latest Kubectl**

```
$curl -LO"https://dl.k8s.io/release/$(curl -L -shttps://dl.k8s.io/release/stable.txt)/bin/linux/amd64
#make the downloaded file executable
$chmod +x kubectl
#Move the executable to the /usr/local/bin
$sudo mv kubectl /usr/local/bin
```

      ii. **Installing Latest Kops**

```
$curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos

#Make the binary executable
$chmod +x kops-linux-amd64

#Move the executable to /usr/local/bin
$sudo mv kops-linux-amd64 /usr/local/bin/kops
```

h. **Create Kubernetes Clusters with 1 Master and 1 Working Node for each Zones respectively for Higher Availability.**

```
$export KOPS_STATE_STORE="s3://pki.atlasqa.co.uk"
$export MASTER_SIZE=${MASTER_SIZE:-m4.large}
$export NODE_SIZE=${NODE_SIZE:-m4.large}
$export ZONES="eu-west-1a,eu-west-1b,eu-west-1c"

$kops create cluster pki.atlasqa.co.uk --node-count 3 --zones $ZONES --node-size $NODE_SIZE --master-size
```

i. **Update the Cluster using the following command.**

```
$kops update cluster --name pki.atlasqa.co.uk--yes --admin
```

j. **Wait for the Cluster to get ready and check the status with the following command.**

```
$kops validate cluster --name pki.atlasqa.co.uk
```

Now your Kubernetes cluster is ready with 3 Working nodes and 3 master running in us-east-1a,us-east-1b,us-east-1c regions respectively.

k. **Install cert-manager and its CRD's**

l. **Add and update the Jetstack Helm repository**

```
$helm repo add jetstack https://charts.jetstack.io --force-update
```

m. **Install the CRD's(Custom Resource Definition) of Certmanager using the following command**

```
$kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.13.3/cert-manager.crd
```

n. **Install the Latest cert-manager using helm**

```
$helm install cert-manager jetstack/cert-manager --namespace cert-manager --create-namespace --version v1
```

o. **Now, Install the GlobalSign's Certmanager-Atlas Issuer CRD. Once it is installed, then it is ready to handle Atlas Certificate requests.**

```
$kubectl apply -f https://github.com/globalsign/atlas-cert-manager/releases/download/v0.0.1/install.yaml
```

p. **Label the cert-manager namespace to disable resource validation**

```
$kubectl label namespace cert-manager certmanager.k8s.io/disable-validation=true
```

3. **Now Install and Configure Nginx-Ingress Controller for your Kubernetes Clusters**

```
$helm upgrade --install ingress-nginx ingress-nginx --repohttps://kubernetes.github.io/ingress-nginx--namespac

$kubectl get svc -n cert-manager
```

4. **Create A records in your Route 53 to the Hosted Zone for the below created Load Balancer IP(Here the cluster IP is 10.100.96.178)**

```
ubuntu@ip-[redacted]:~$ kubectl get svc -n cert-manager
NAME                                TYPE           CLUSTER-IP       EXTERNAL-IP                                                              POR
T(S)                    AGE
cert-manager                        ClusterIP      10.100.241.61    <none>                                                                   940
2/TCP                   94s
cert-manager-webhook                ClusterIP      10.100.246.181   <none>                                                                   443
/TCP                    94s
ingress-nginx-controller            LoadBalancer   10.100.96.178    a99d49e4a1a62409689a287279f59e79-2014922034.eu-west-1.elb.amazonaws.com   80:
32365/TCP,443:32754/TCP 21s
ingress-nginx-controller-admission  ClusterIP      10.100.219.192   <none>                                                                   443
/TCP                    21s
```

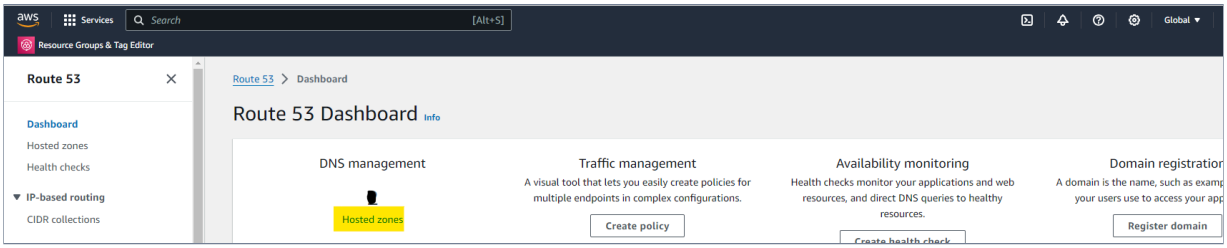
As soon as the ingress-nginx-controller get the EXTERNAL-IP value with extension *.eu-west-1.elb.amazonaws.com, Add this value as A record into hosted zone. It would be in the sync within 60sec.

Note:- Before creating the hosted zone kindly make sure you have the valid domain.

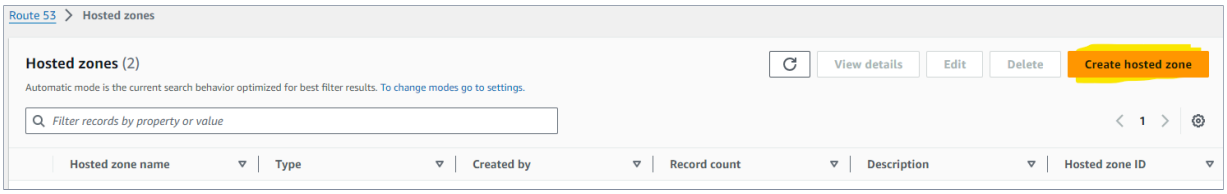
a. **To create A records into your Route 53, create one Hosted Zone**

Go to https://us-east-1.console.aws.amazon.com/route53/v2/home?region=eu-west-1#Dashboard and click on "Hosted zones".

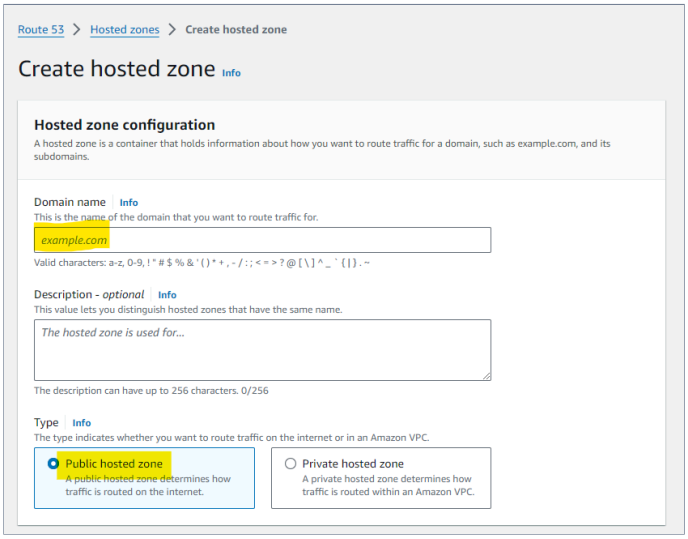
b. **Create a hosted zone**



c. **Click on "Create hosted zone"**



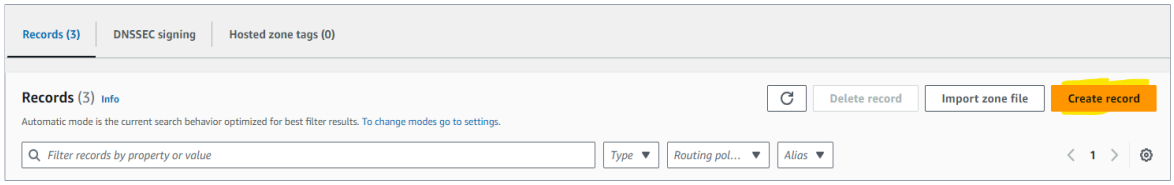
d. **Enter the name followed by your actual domain name and make sure the "Public hosted zone" should be selected:**



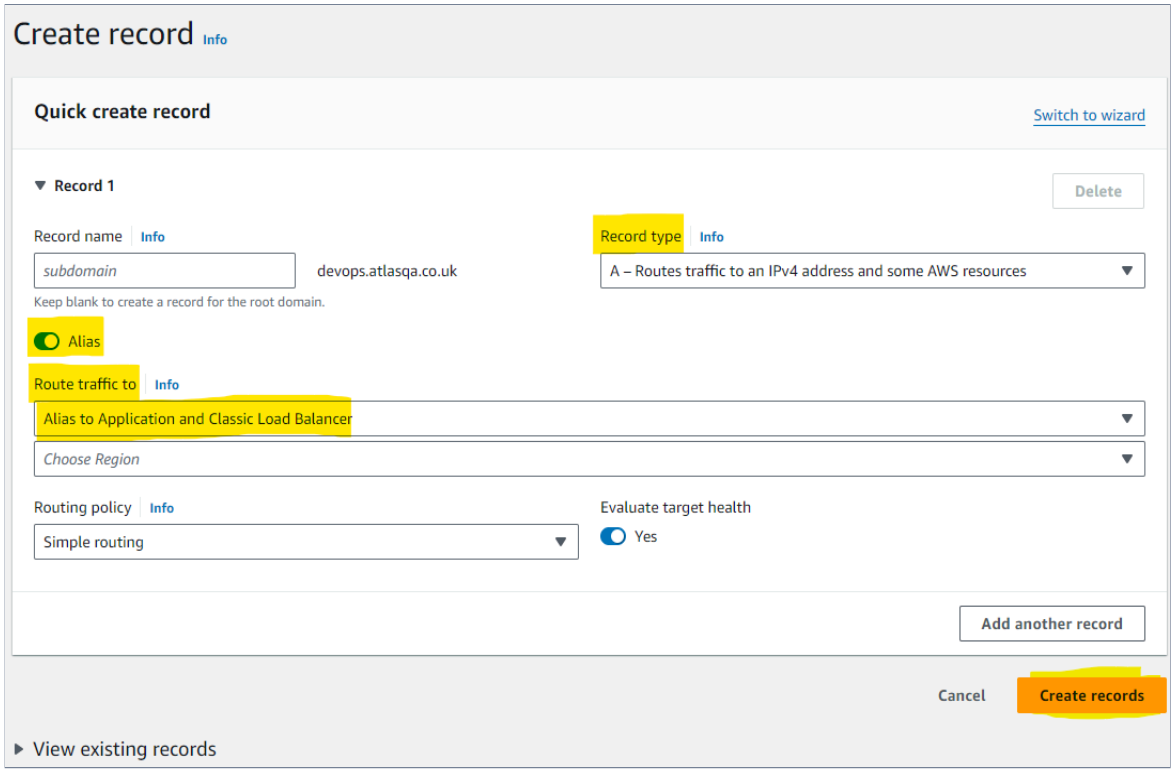
e. **After creating the hosted zone, you would get some NS record along with SOA record. Now add the NS records into your domain registrar**

f. **After adding the NS into domain registrar your hosted zone is now ready to accept traffic, Now you can create the A record into the hosted zone:**

   i. Into your hosted zone, Click on "Create record"



   ii. On the next screen make sure that Record Type is "A" and "Alias" are selected. Also make sure that "Route traffic to" "Alias to Application and Classic Load Balancer is selected". After selecting the required fields click on "Create records":



iii. The record would be created and it would take around 60sec to get in the sync.

5. **Create GlobalSign Issuer to issue a TLS certificate for your Ingress using the following steps**:-

   a. **Create a secret to store the GlobalSign's ATLAS account api_key, secrets along with mTLS and private key(You can get these API credentials from GlobalSign's Team)**

```
$kubectl create secret generic issuer-credentials --from-literal=apikey=$API_KEY --from-literal=apisecret
```

   b. **Create an Issuer of GlobalSign.**

issuer.yaml

```
cat <<EOF | kubectl apply -f -
apiVersion: hvca.globalsign.com/v1alpha1
kind: Issuer
metadata:
  name: gs-issuer
  namespace: cert-manager
spec:
  authSecretName: "issuer-credentials"
  url: "https://emea.api.hvca.globalsign.com:8443/v2"
EOF
```

   c. **Create Certificate Resource with the following Configuration**

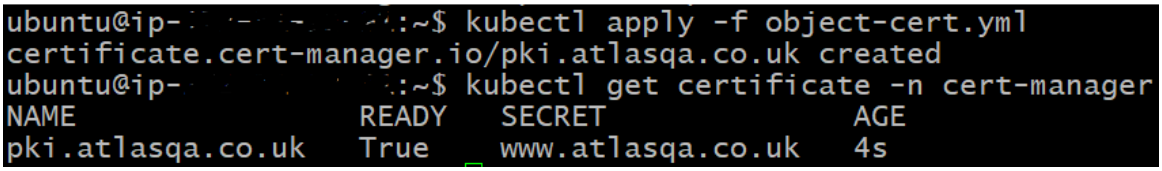
cert.yaml

```
cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: pki.atlasqa.co.uk
  namespace: cert-manager
spec:
  # Secret names are always required.
  secretName: www.atlasqa.co.uk

  duration: 2160h # 90d
  renewBefore: 360h # 15d
  subject:
  #  organizations:
  #    - jetstack
```

```
  # The use of the common name field has been deprecated since 2000 and is
  # discouraged from being used.
  commonName: pki.atlasqa.co.uk
  isCA: false
  privateKey:
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  usages:
    - server auth
    #- client auth
  # At least one of a DNS Name, URI, or IP address is required.
#  dnsNames:
#    -
#www.atlasqa.co.uk
  # Issuer references are always required.
  issuerRef:
    name: gs-issuer
    # We can reference ClusterIssuers by changing the kind here.
    # The default value is Issuer (i.e. a locally namespaced Issuer)
    kind: Issuer
    # This is optional since cert-manager will default to this value however
    # if you are using an external issuer, change this to that issuer group.
    group: hvca.globalsign.com
EOF
```

d. **At times the certificate object can take couple of seconds to become READY.**



6. **Securing Nginx ingress resource by the below configuration:**

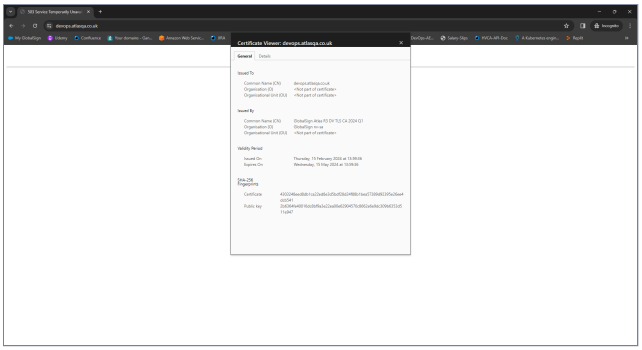
ingress.yaml

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  namespace: cert-manager
  annotations:
    cert-manager.io/issuer: GS-issuer
    kubernetes.io/ingress.class: nginx
spec:
  tls:
    - hosts:
        - pki.atlasqa.co.uk
      secretName: www.atlasqa.co.uk
  rules:
    - host: pki.atlasqa.co.uk
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: example-service
                port:
                  number: 80
EOF
```

7. **The ingress resource that has been created could take up to 1min to get the load balancer URL as ADDRESS.**

```
ubuntu@ip-          ':~$ kubectl apply -f ingress.yml
Warning: annotation "kubernetes.io/ingress.class" is deprecated, please use 'spec.ingressClassName' instead
ingress.networking.k8s.io/nginx-ingress created
ubuntu@ip-              :~$ kubectl get ingress -n cert-manager
NAME            CLASS    HOSTS                ADDRESS                                                                  PORTS     AGE
nginx-ingress   <none>   devops.atlasqa.co.uk   a215bae5711314b3cb9a9bd1556950fb-1850122743.us-east-2.elb.amazonaws.com   80, 443   87s
```

8. **Now that the domain https://pki.atlasqa.co.uk/ has the GS public TLS certificate.**



Demo: Demo video